Atelier Data Science

Deep learning practice 2

Convolutional Neural Networks

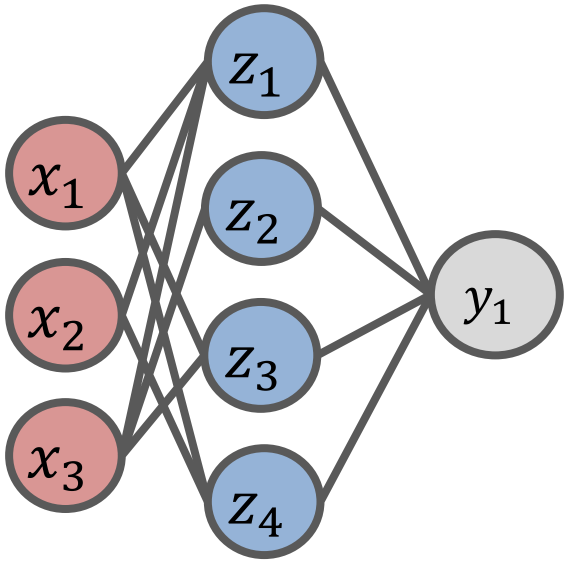
Irina Proskurina

Irina.Proskurina@univ-lyon2.fr

Laboratoire ERIC – Université Lyon 2

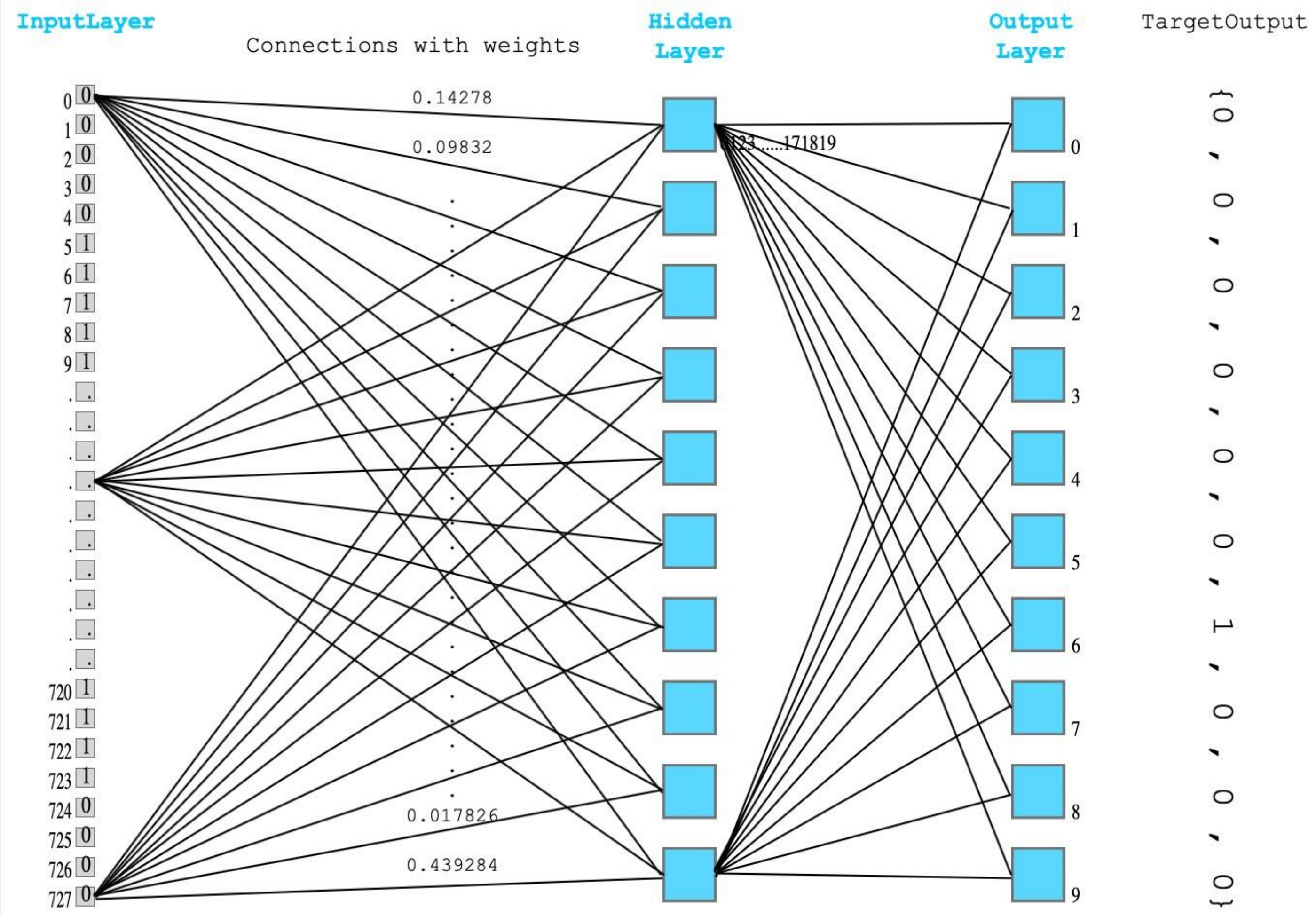
Previous Lesson Recap 1

1) Neural Networks -> no manual feature engineering is needed!
2) Fully connected layers (**torch.nn.Linear**(20, 30))
3) Fully connected neural networks: connect every neuron in one layer to every neuron in the other layer + activation function



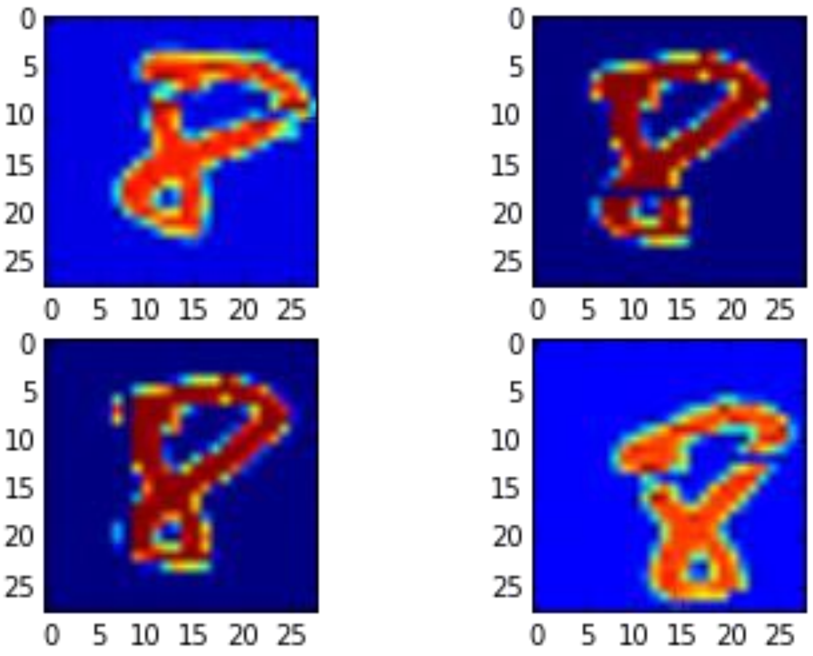
Previous Lesson Recap 2
MNIST

- Each neuron can detect the presence of a specific set of pixels



Previous Lesson Recap 2
MNIST

- If you shift the digit slightly, the neuron will no longer detect its pattern



https://srome.github.io/Jitter,-Convolutional-Neural-Networks,-and-a-Kaggle-Framework/

Number of parameters

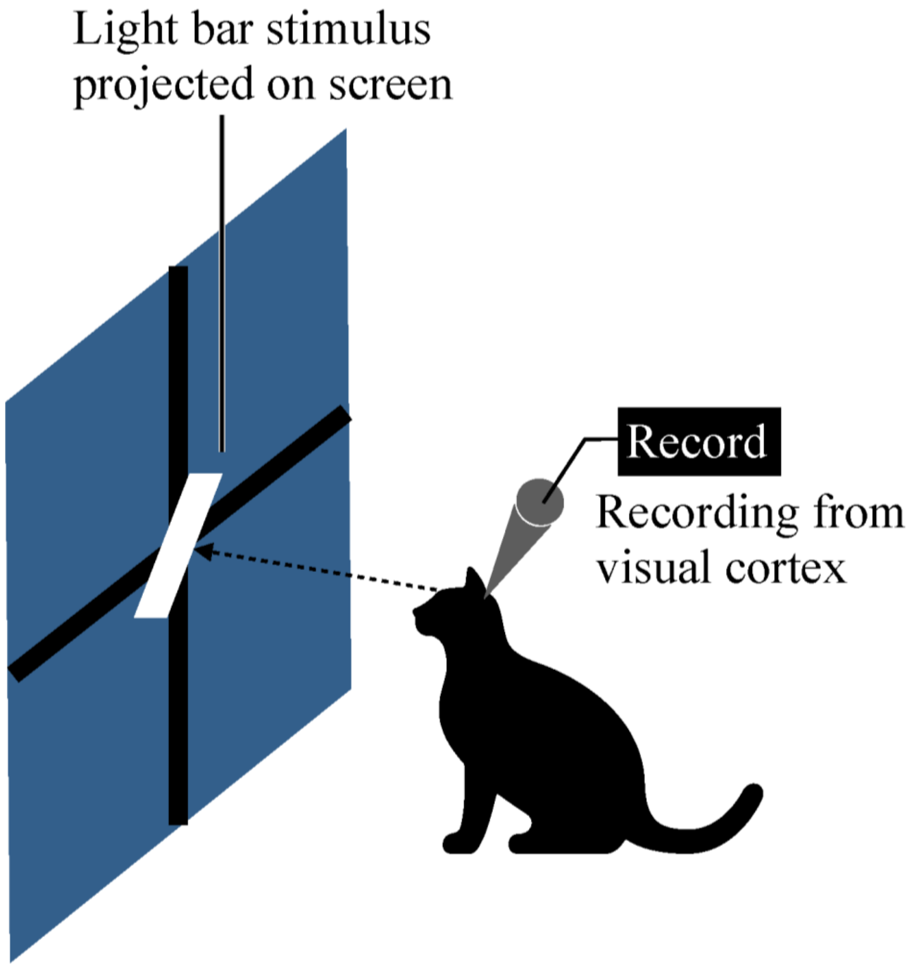
- 784 inputs
- Fully connected layer: 1000 neurons
- Output layer: 10 neurons (one for each class)
- Weights between input and fully connected layers: (784 + 1) * 1000 = 785,000
- Weights between fully connected and output layers: (1000 + 1) * 10 = 10,010

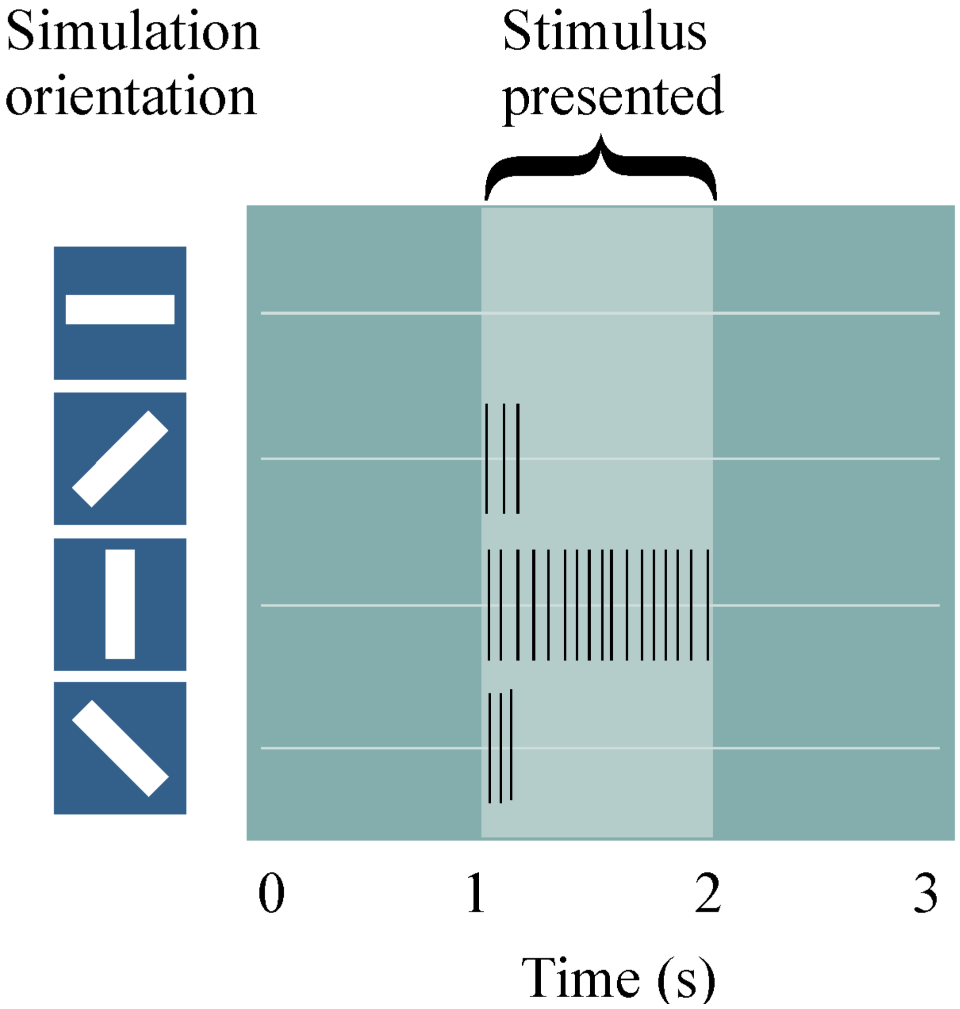
Fully connected neural networks for image classification

- A lot of parameters
- Prone to overfitting
- Do not consider special patterns in images: shifts, slight changes in shape, etc.
- One of the best ways to combat overfitting is to reduce the number of parameters

Convolutional neural network

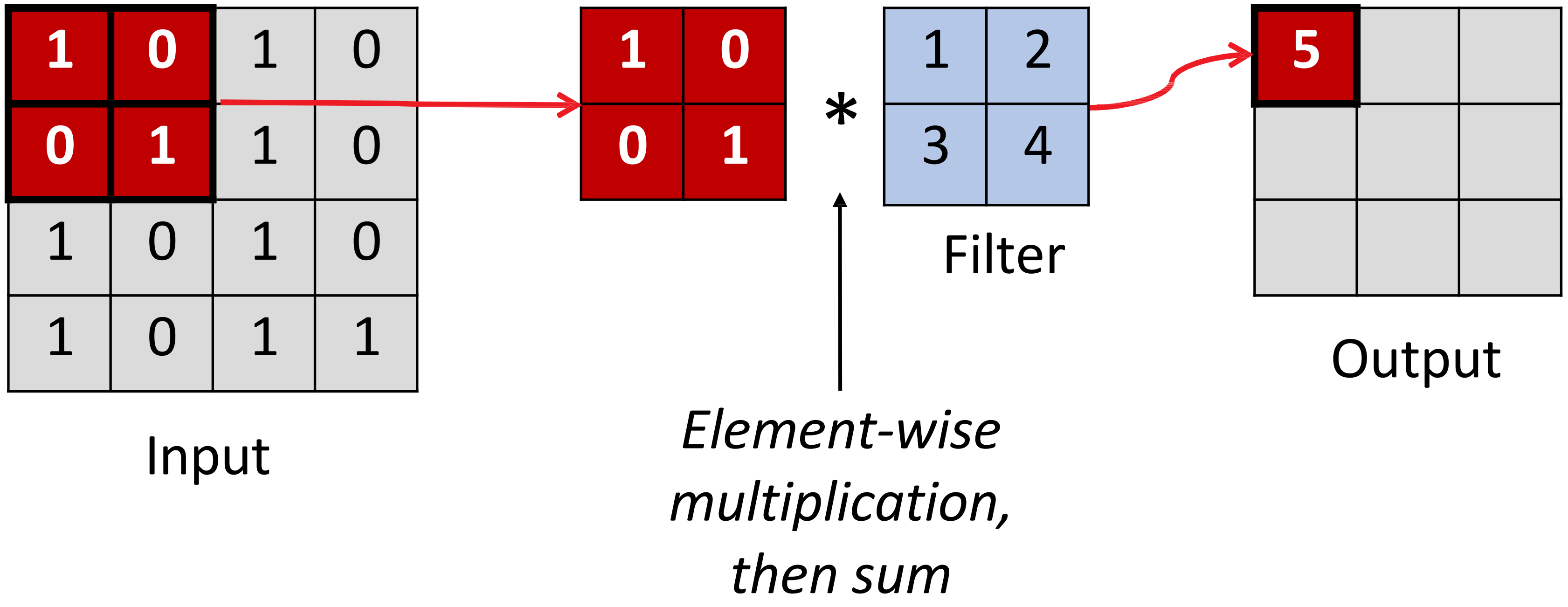
Experiments with the visual cortex





https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/

Convolution



Convolution

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{1}$$

$$\begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{0}$$

$$\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{6}$$

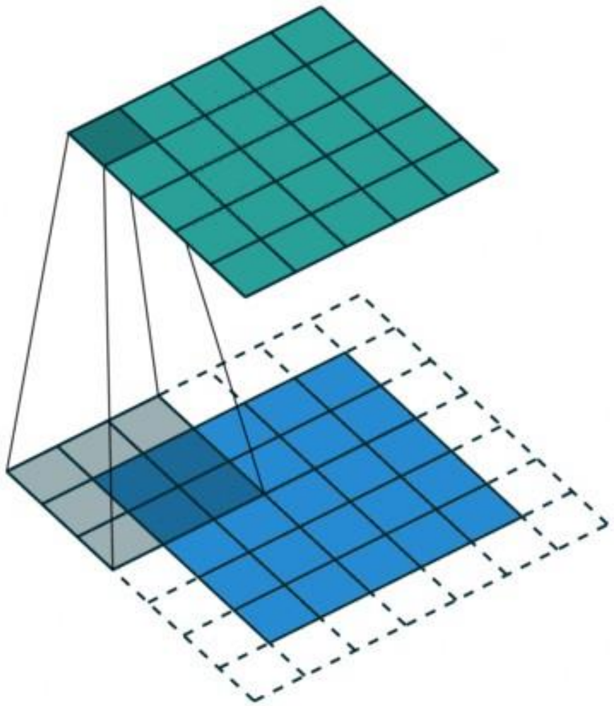
$$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \boxed{10}$$

Convolution

- Detects a pattern in the image, which is defined by a filter
- The stronger the pattern in a particular area of the image, the higher the convolution value will be

Convolution

- The result of the convolution is a new image, where each pixel is a weighted sum of the pixels in the original image



The maximum of convolution is invariant to shifts

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Filter}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}}_{\text{Output}}$$

Max = 2

Same

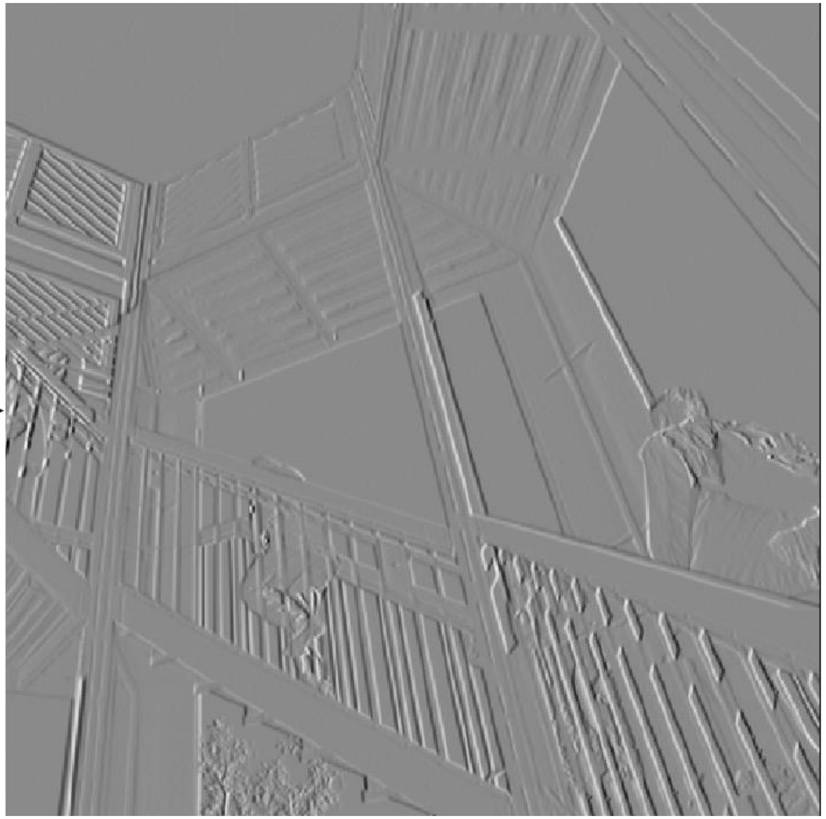
$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Filter}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Output}}$$

Max = 2

Convolutions in computer vision

$$\longrightarrow \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \longrightarrow$$

Horizontal Sobel kernel



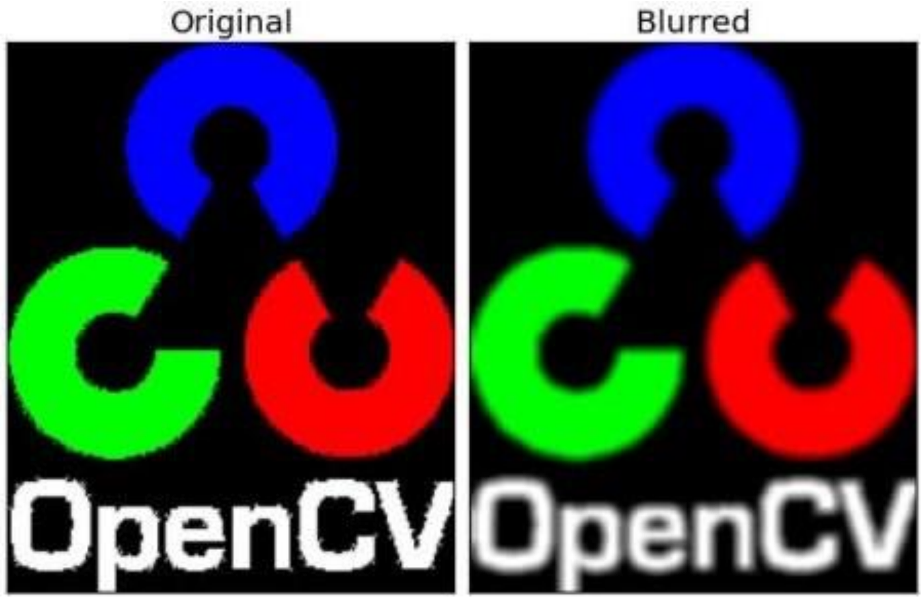
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolutions in computer vision

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

Convolutions in computer vision



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

Convolution

$$Im^{out}(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \left(K(i,j) Im^{in}(x+i, y+j) + b \right)$$

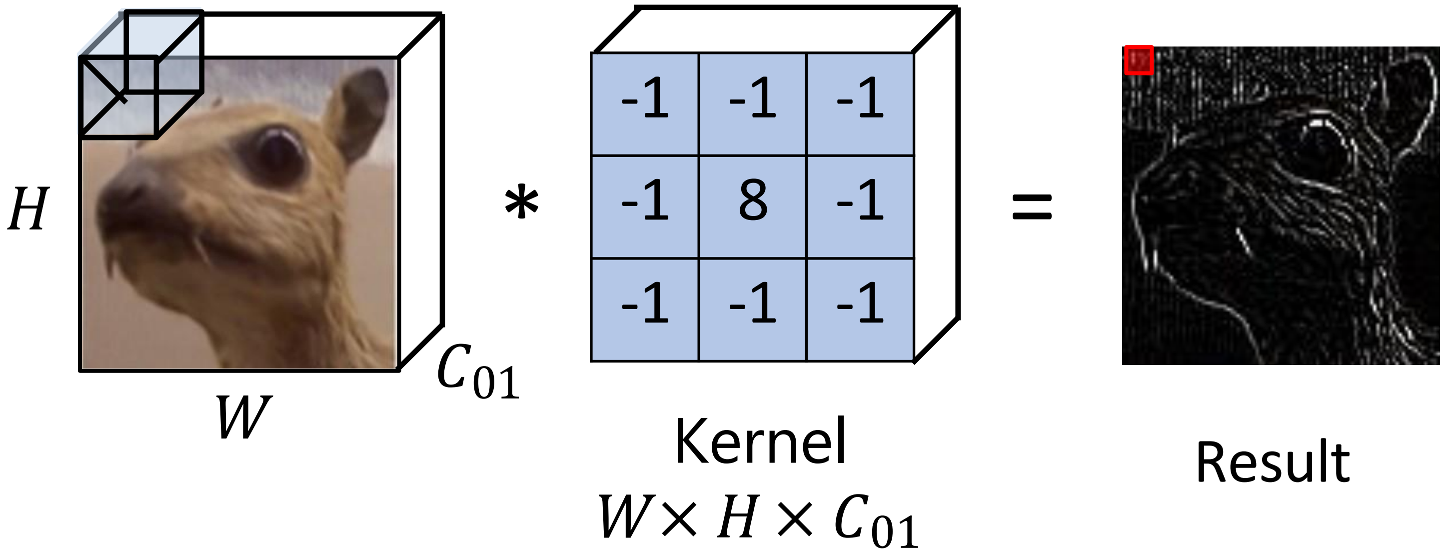
- A pixel in the resulting image depends only on a small set of the input image (local connectivity)
- The weights (kernel values) are the same for all pixels in the output image (shared weights)

Convolution

- Typically, the original image is in color, implying it has multiple channels (R, G, B)
- Let's take this into account in the formula:

$$Im^{out}(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} \left(K(i,j,c) Im^{in}(x+i, y+j, c) + b\right)$$

Convolution



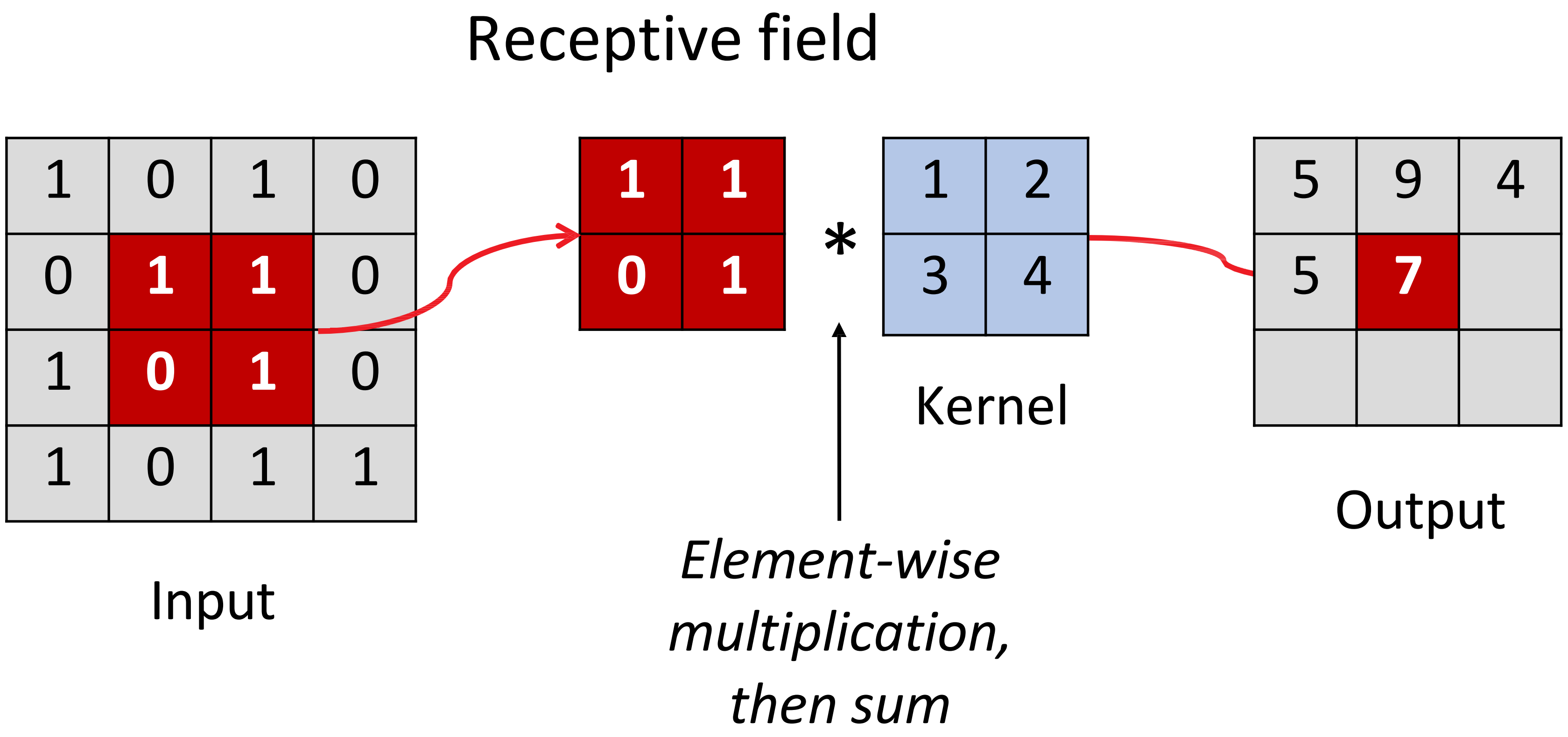
Number of parameters

$$Im^{out}(x,y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} \left(K(i,j,c)Im^{in}(x+i,y+j,c) + b_t\right)$$

- Kernel **parameters**
- $\left((2d+1)^2 * C + 1\right) * T$

Receptive field

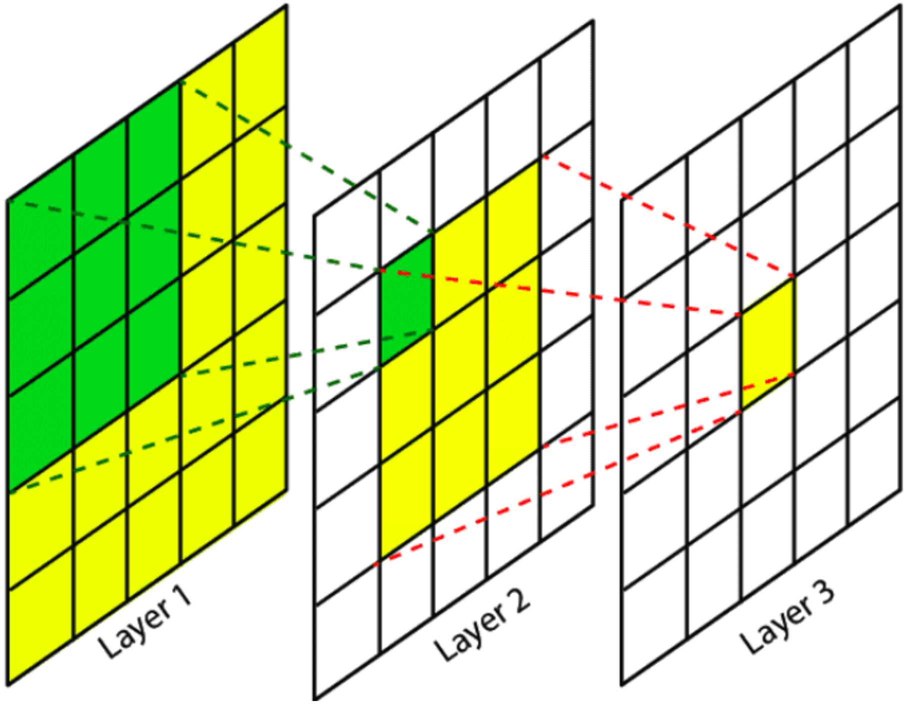
Convolution



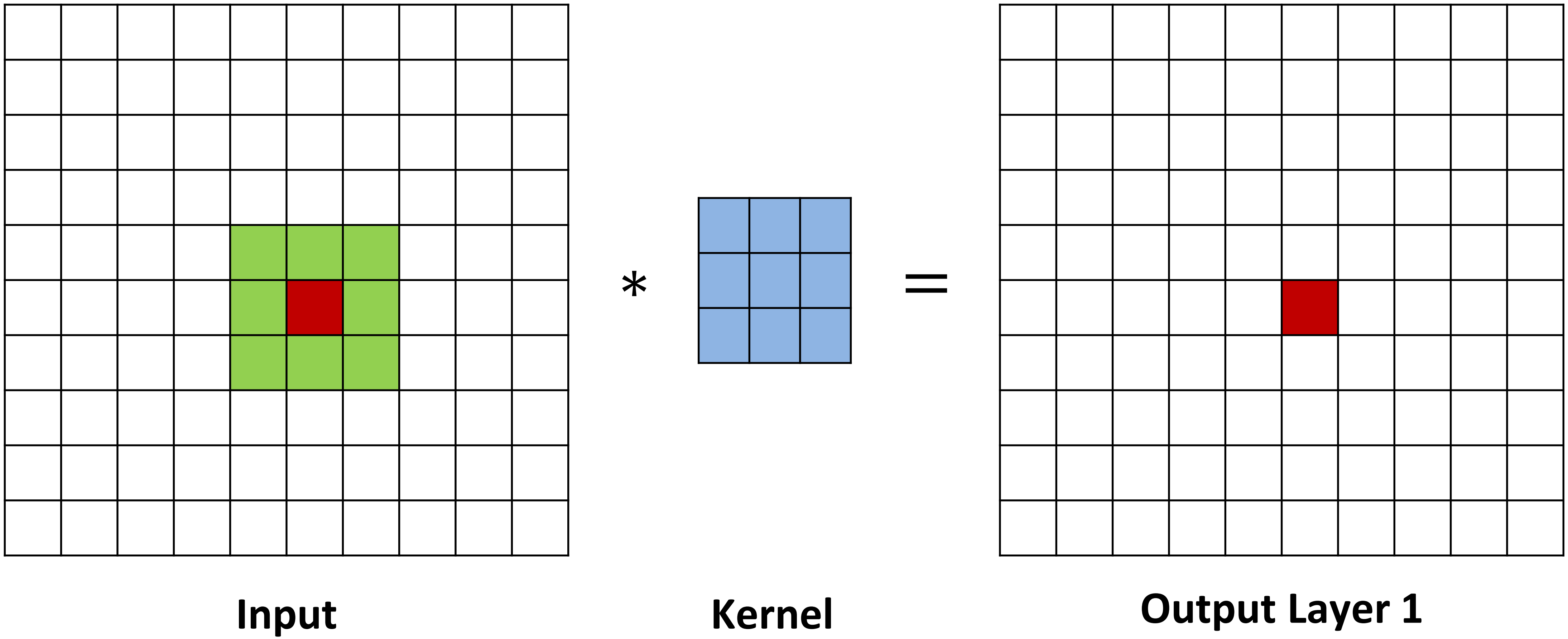
Receptive field

- Consider a pixel in the output image
- What part of the input image influences the value in this output pixel?

Receptive Field (RF)
the size of the region
in the input that
produces the feature

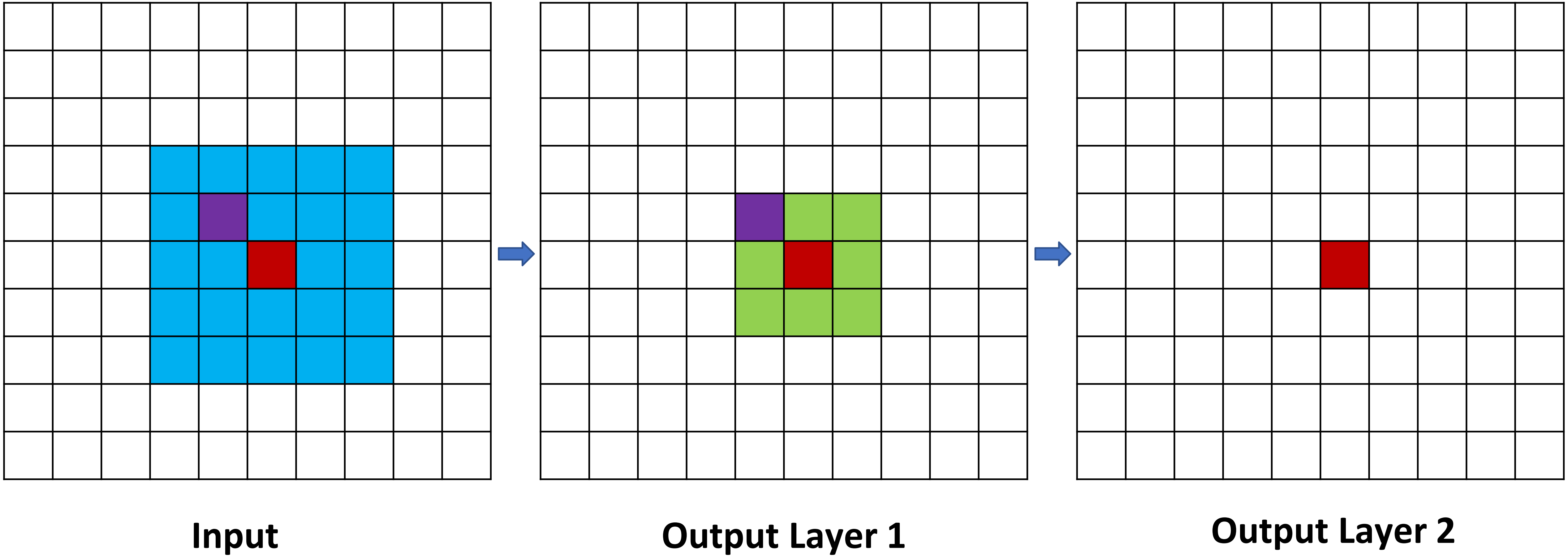


Receptive field



Receptive field: 3 x 3

Receptive field



Receptive field: 5 x 5

Receptive field

Receptive field for 3 x 3 convolution:

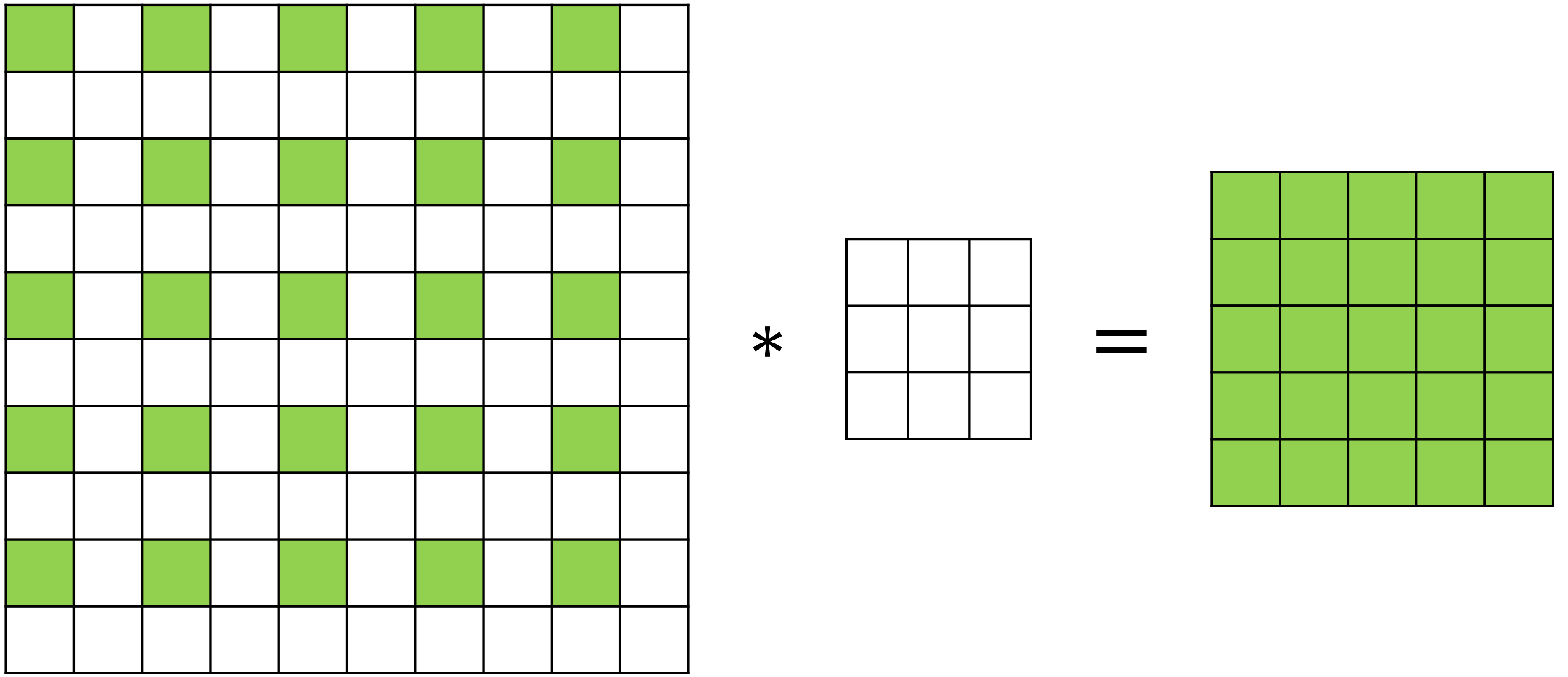
- After 1 convolutional layer: 3 x 3
- After 2 convolutional layers: 5 x 5
- After 3 convolutional layers: 7 x 7

We need a lot of layers if the image size is 512 x 512!

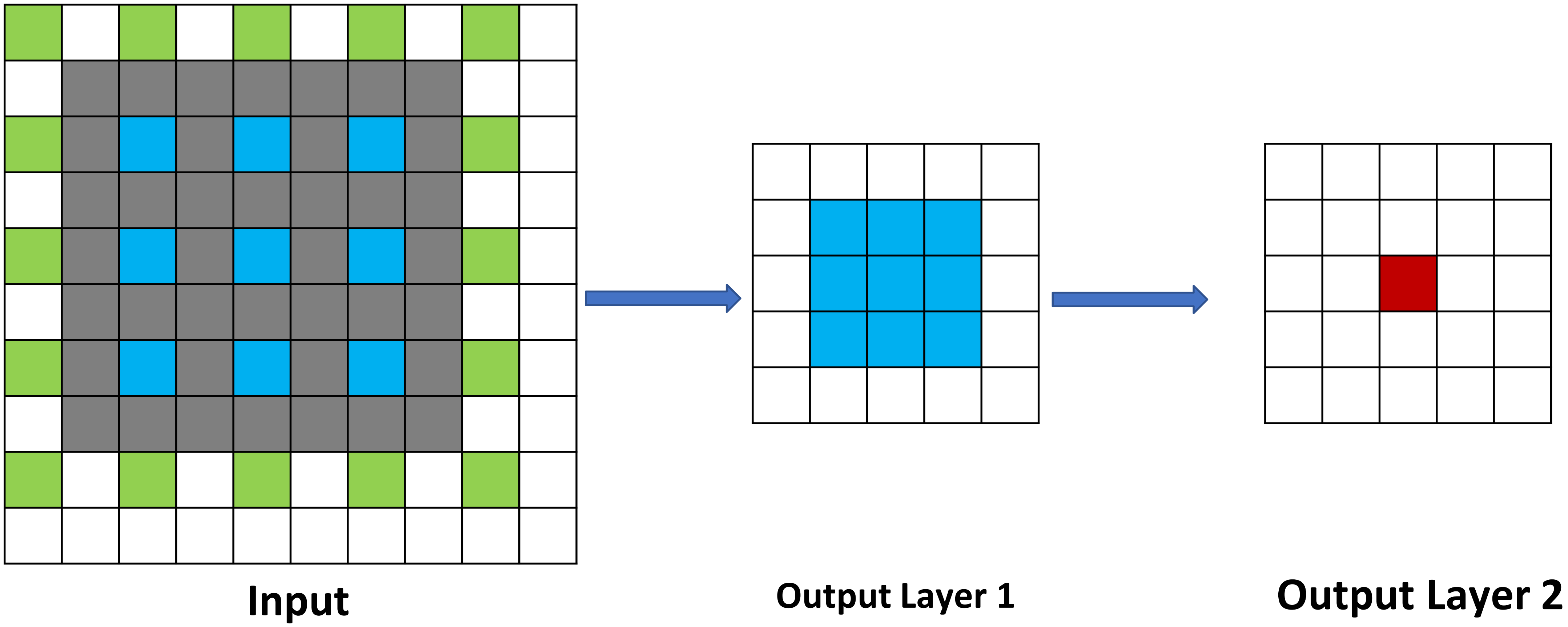
Strides

$s = 2$

Stride:
the number of pixels by which we move the filter across the input image



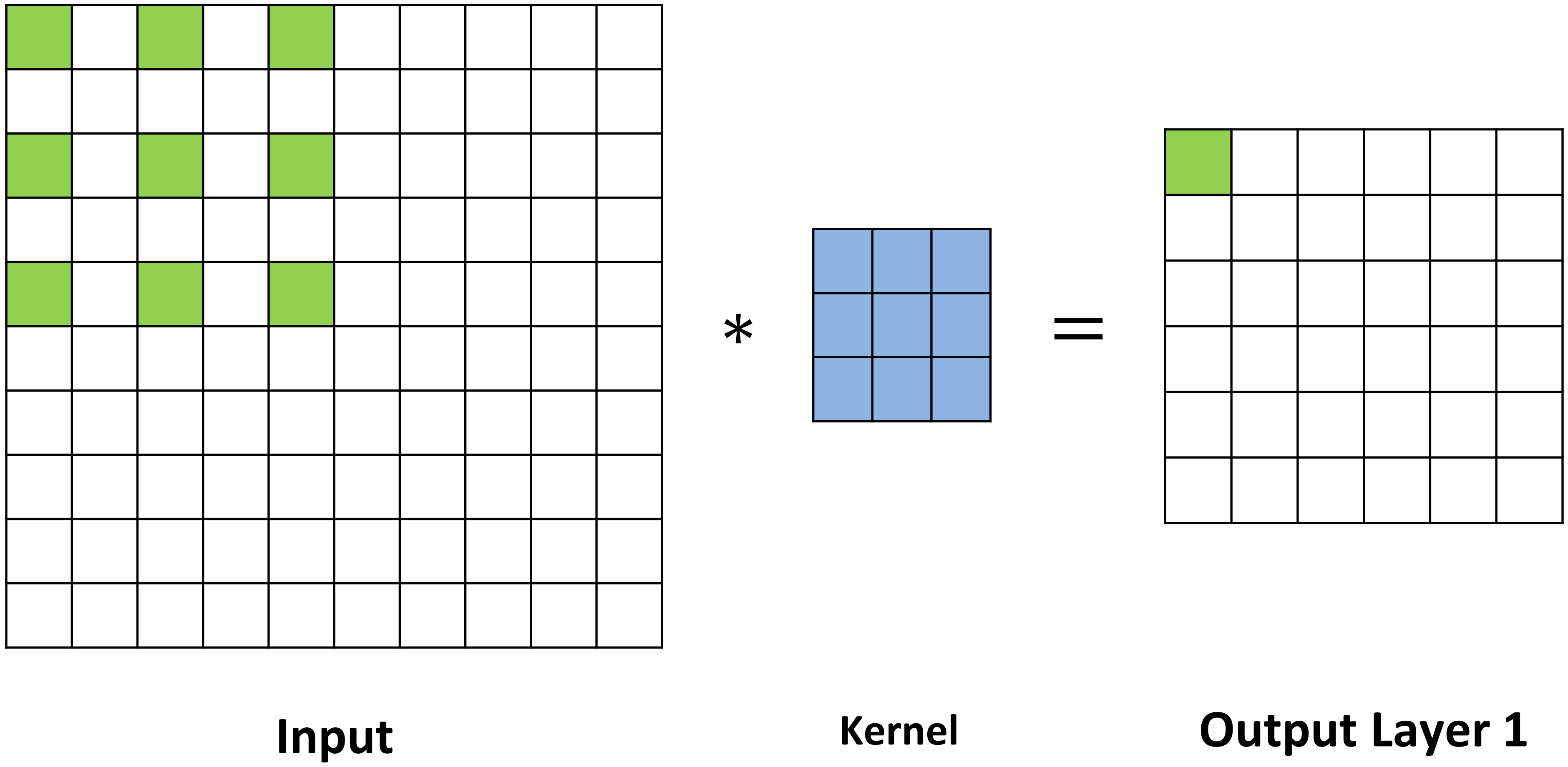
Strides



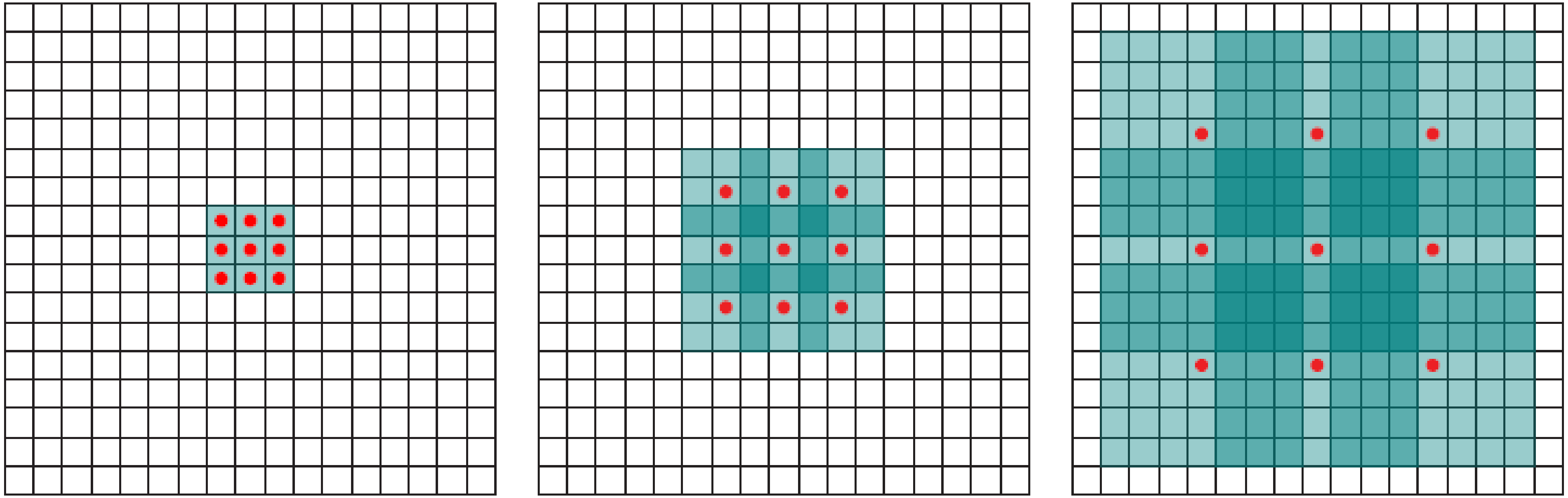
Receptive field: 7 x 7

Dilated convolutions

$l = 2$



Dilated convolutions



$l = 1$ $\qquad$ $l = 2$ $\qquad$ $l = 4$

https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5

The maximum of convolution is invariant to shifts

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Filter}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}}_{\text{Output}}$$

Max = 2

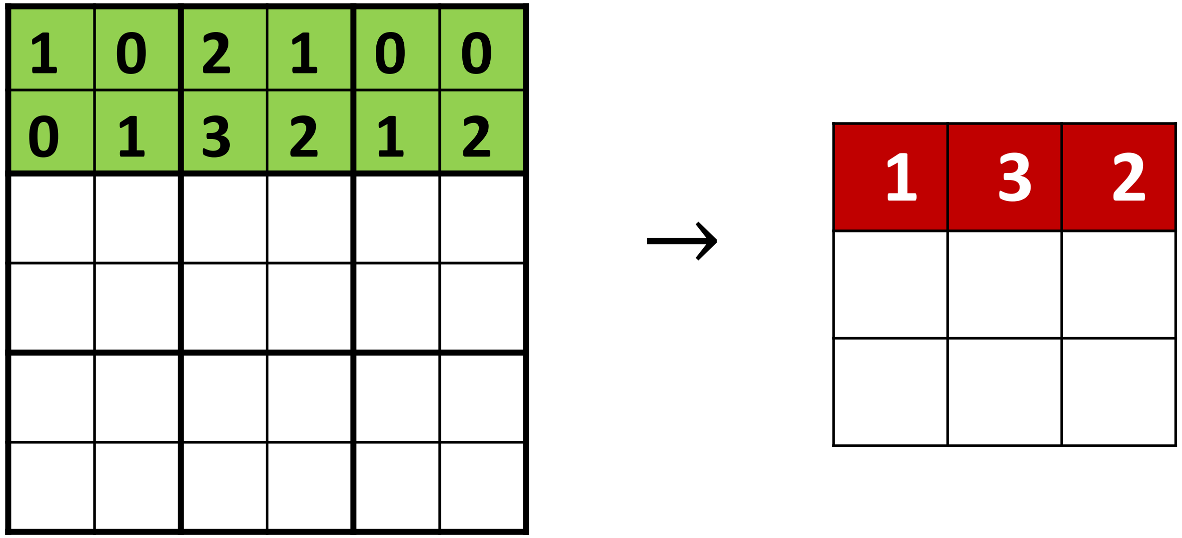
Same

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Filter}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Output}}$$

Max = 2

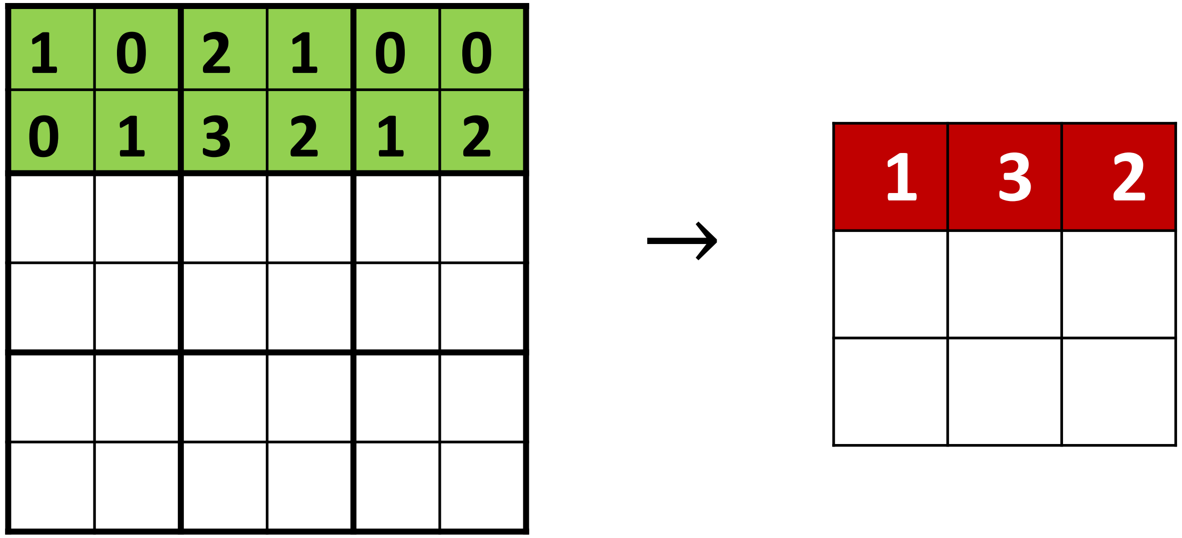
Pooling

- Max-pooling with kernel 2x2



Pooling

- Splits the image into $n \times m$ sections applying some function (usually max)
- Significantly reduces the size of the image (which means it increases the receptive field in the subsequent layers)
- Has no parameters

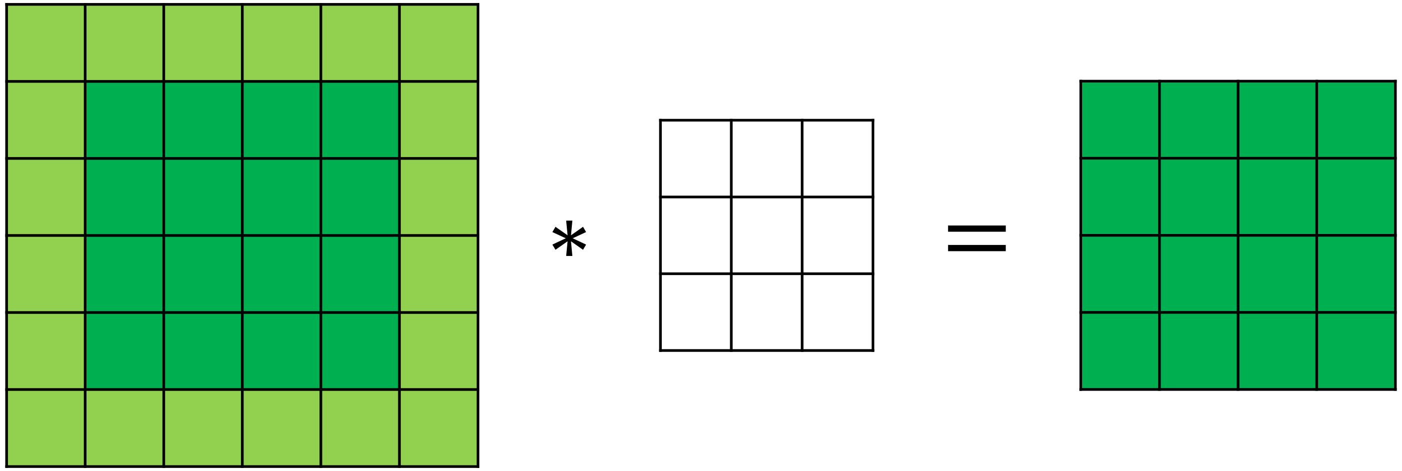


Why we need to know all this?

- It is important to ensure that the last convolutional layer has a receptive field relative to the size of input image

Convolution

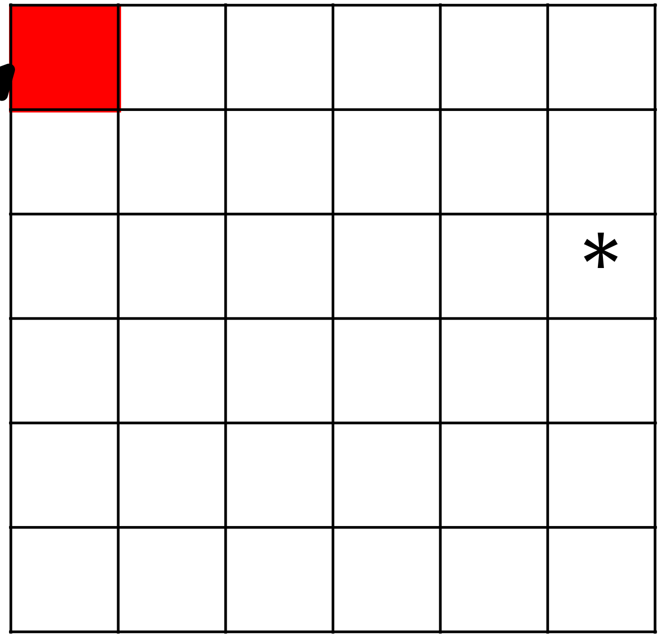
- If you apply convolution, the output image will be smaller than the input



Valid padding (= No padding)

- When applying convolution to image, the pixels at the edges have less impact on the output

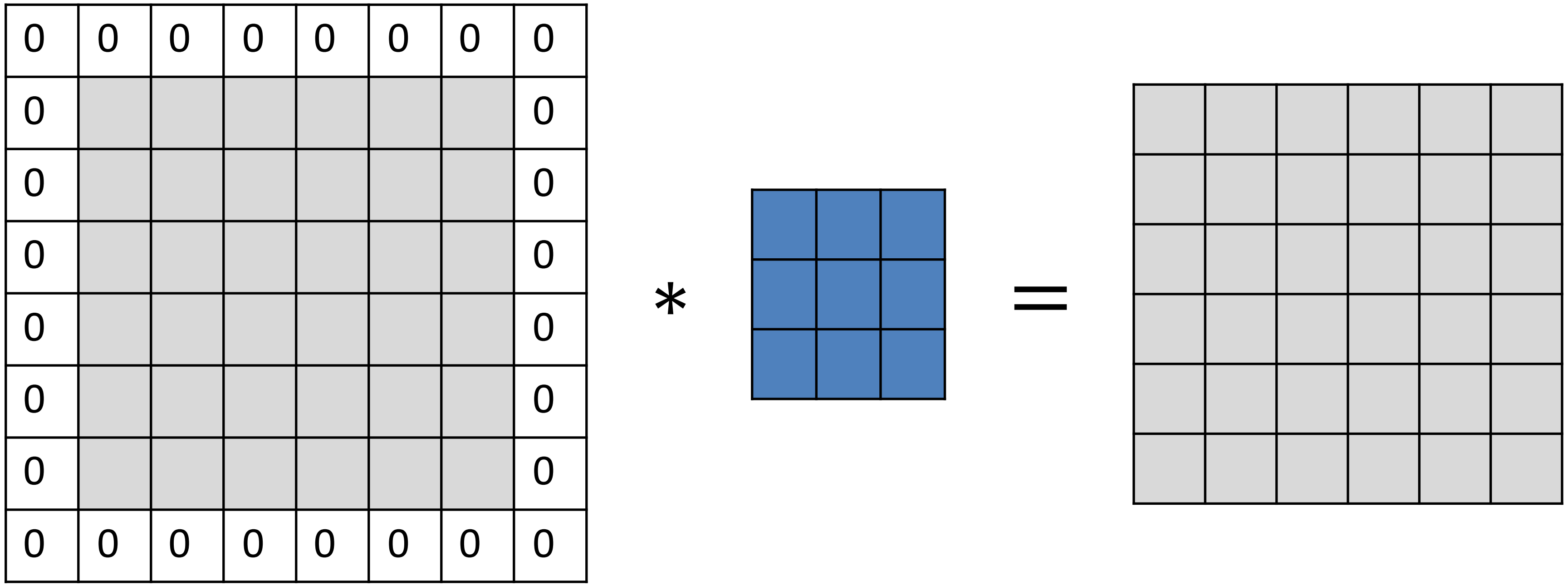
We will not see that the filter has a good response when placing the center at this pixel



*

0	0	1
0	0	1
1	1	1

Zero padding

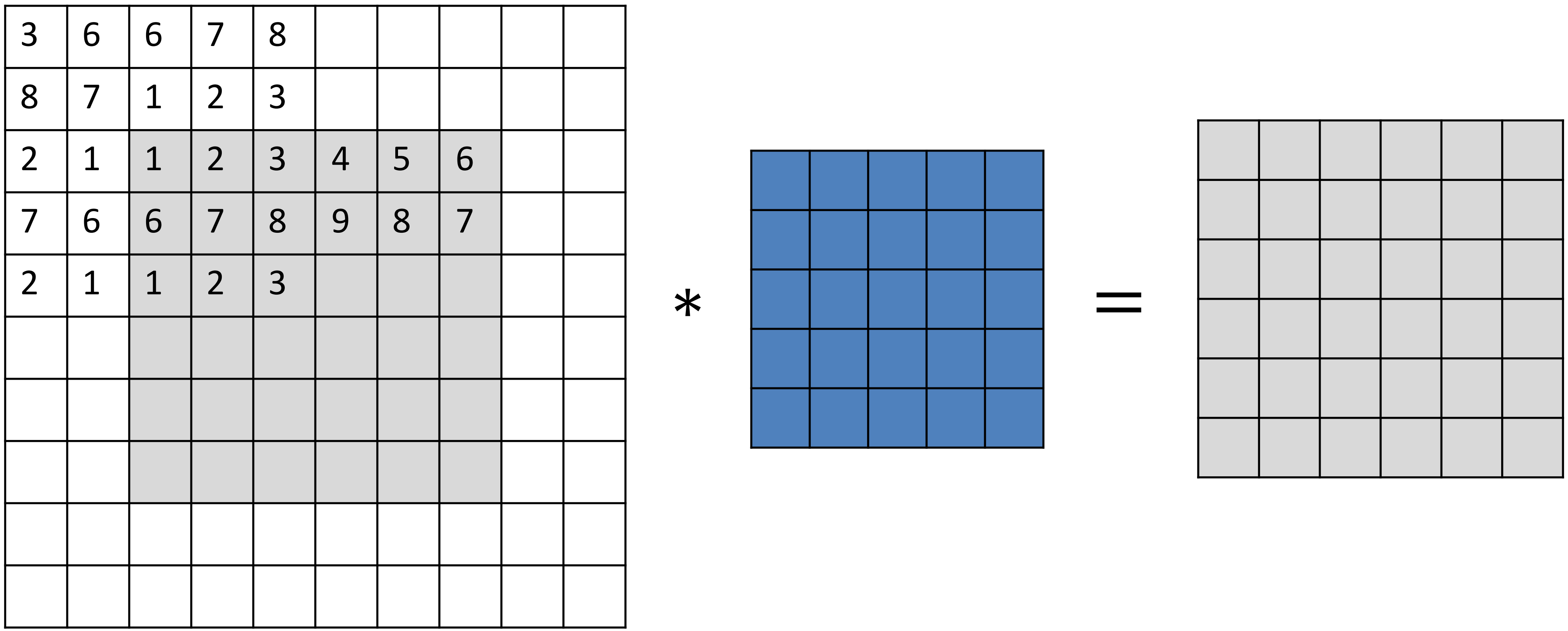


Zero padding

- We add zeros along the boundaries so that the convolution calculated after this in valid mode gives an image of the same size as the original one
- There is a risk that the model will learn to understand where the edges are in the image - we may lose invariance

Reflection padding

- Pad input data symmetrically with a reflection of its own values



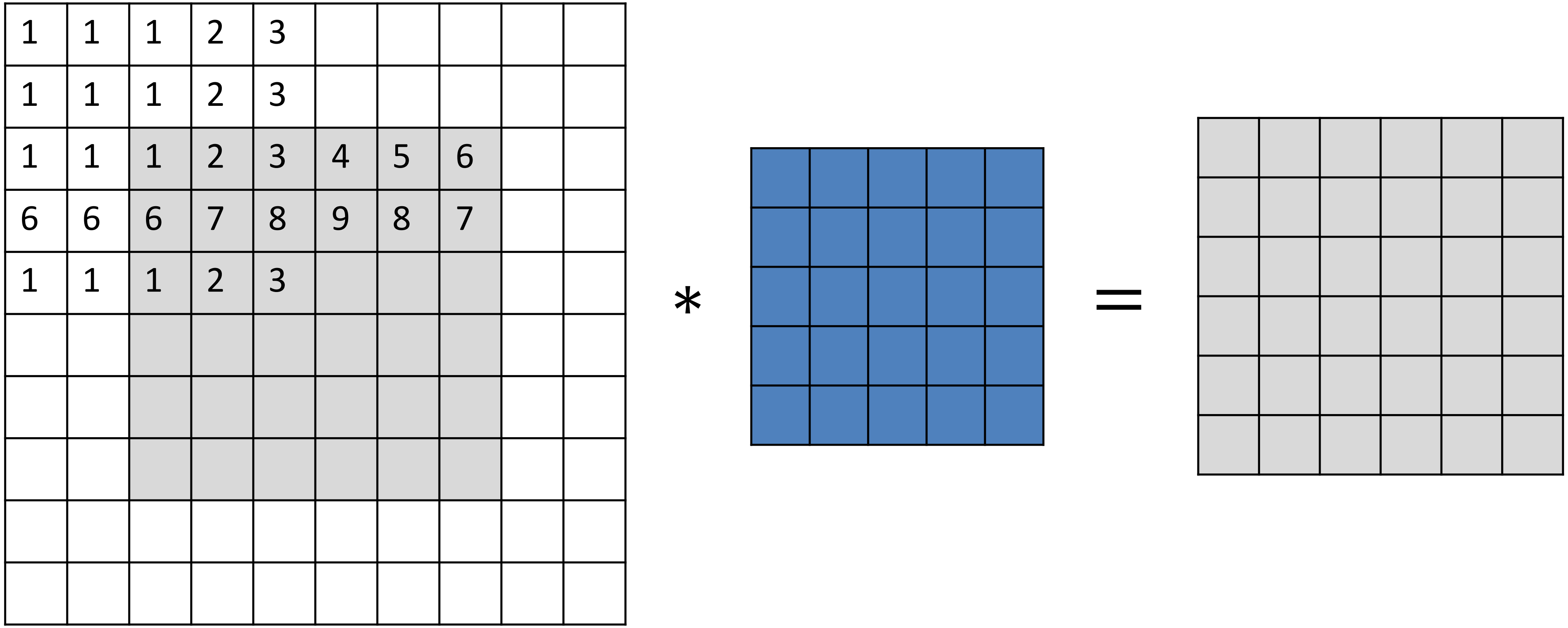
Reflection padding

- Can't easily find image edges
- But now the model can begin to find specular reflections and select filters for them

3	6	6	7	8					
8	7	1	2	3					
2	1	1	2	3	4	5	6		
7	6	6	7	8	9	8	7		
2	1	1	2	3					

Replication padding

- Pad with replicated the values at the boundary



Replication padding

- The pixel on the border is equal to the nearest pixel from the image
- The model can still adjust to the patterns that arise from such padding

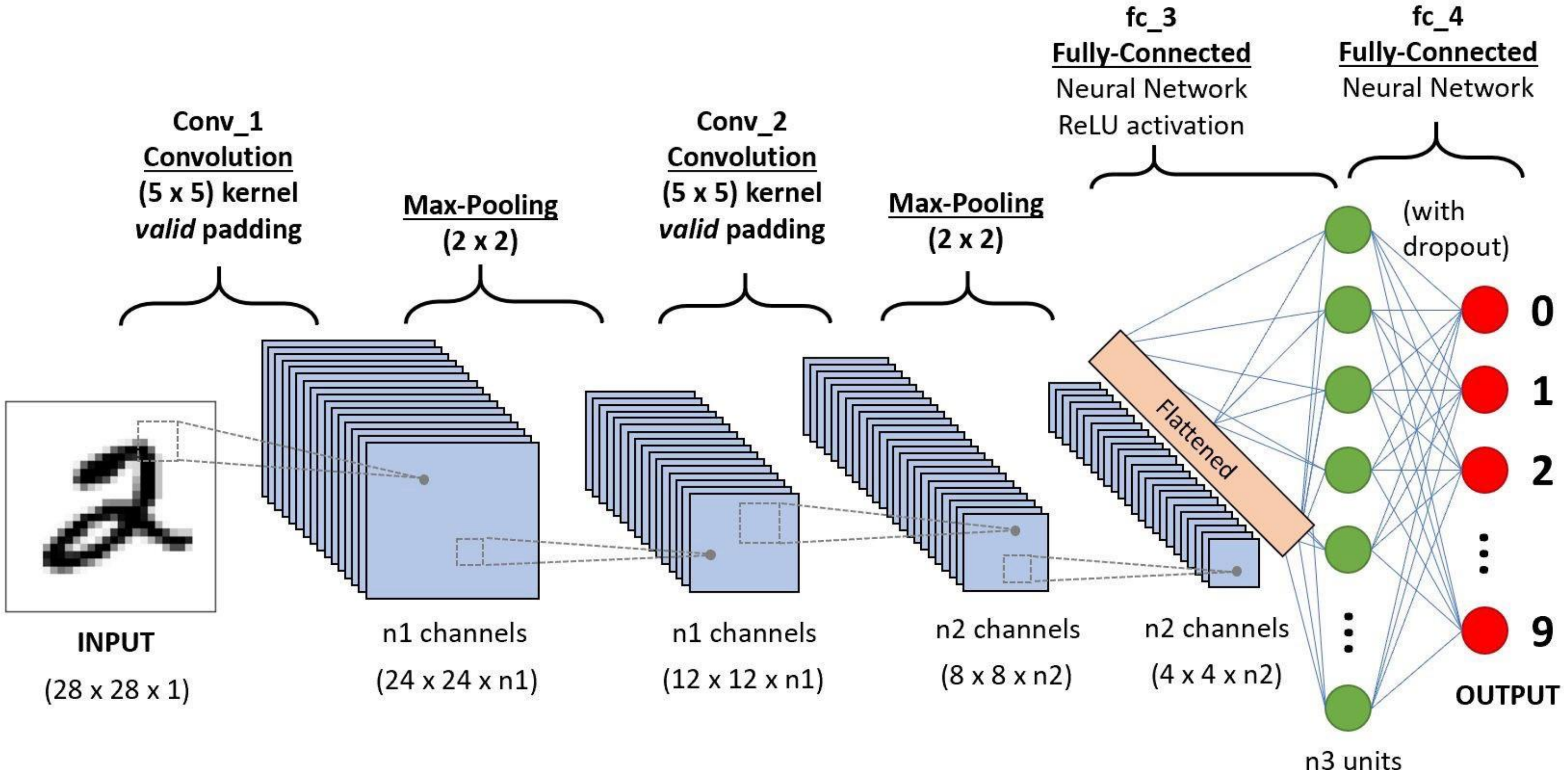
1	1	1	2	3					
1	1	1	2	3					
1	1	1	2	3	4	5	6		
6	6	6	7	8	9	8	7		
1	1	1	2	3					

Summary

- Padding allows you to control the size of the output images
- Padding allows you to take into account objects on the edges
- Different types of padding allow different methods of retraining for edges

Convolutional Neural Networks

Architecture 1

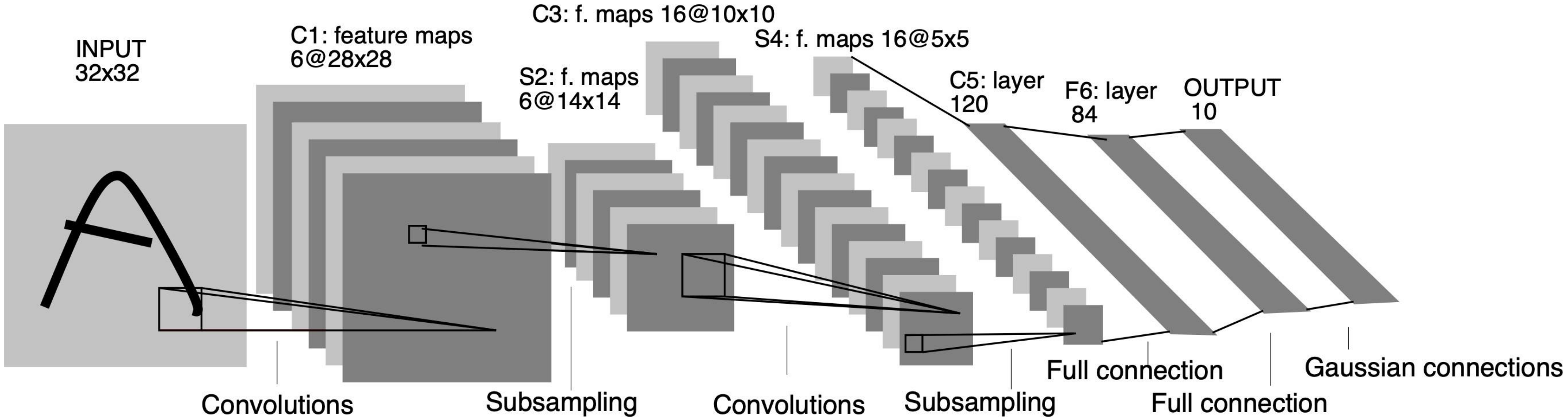


https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Architecture 1

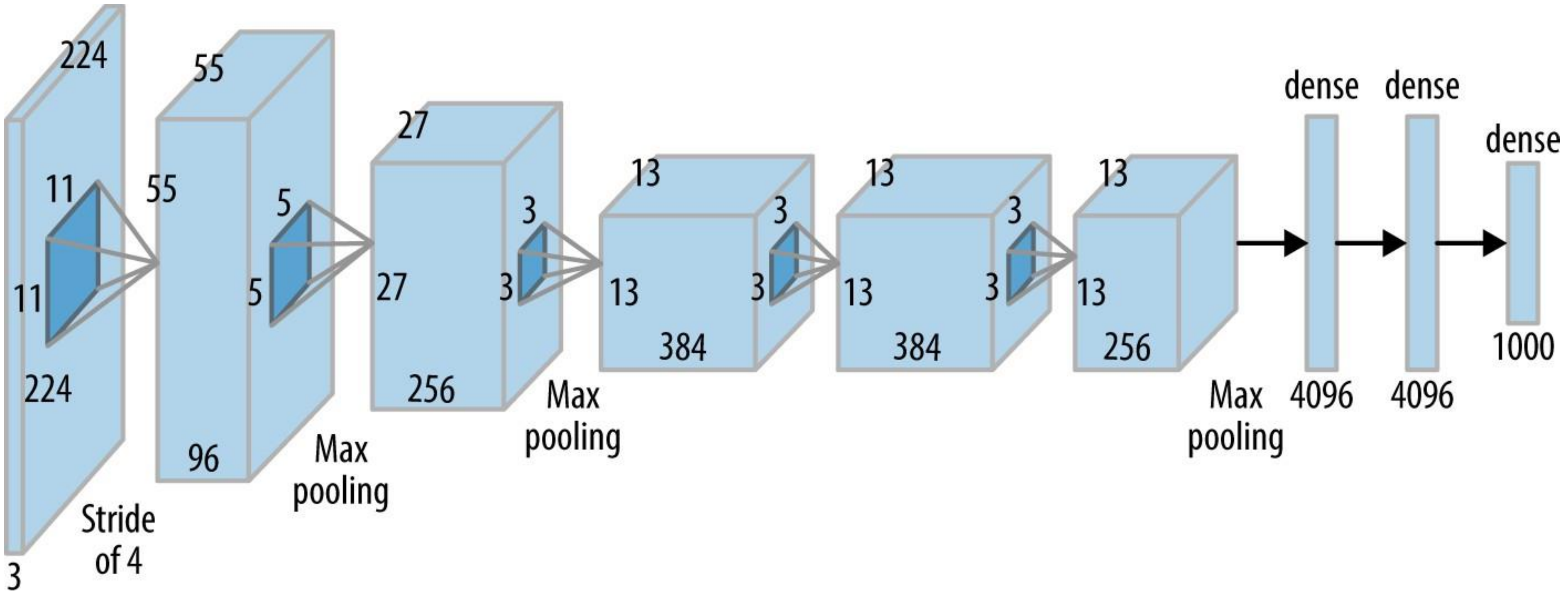
1) Convolution->linear layer>pooling or convolution->non-linear layer
2) Flatten the output
3) Fully-connected layer

Architecture 2: LeNet



http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Architecture 3: AlexNet



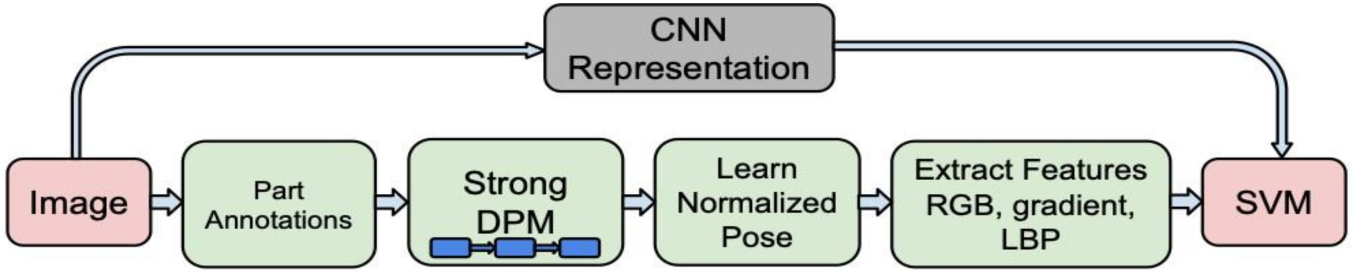
http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

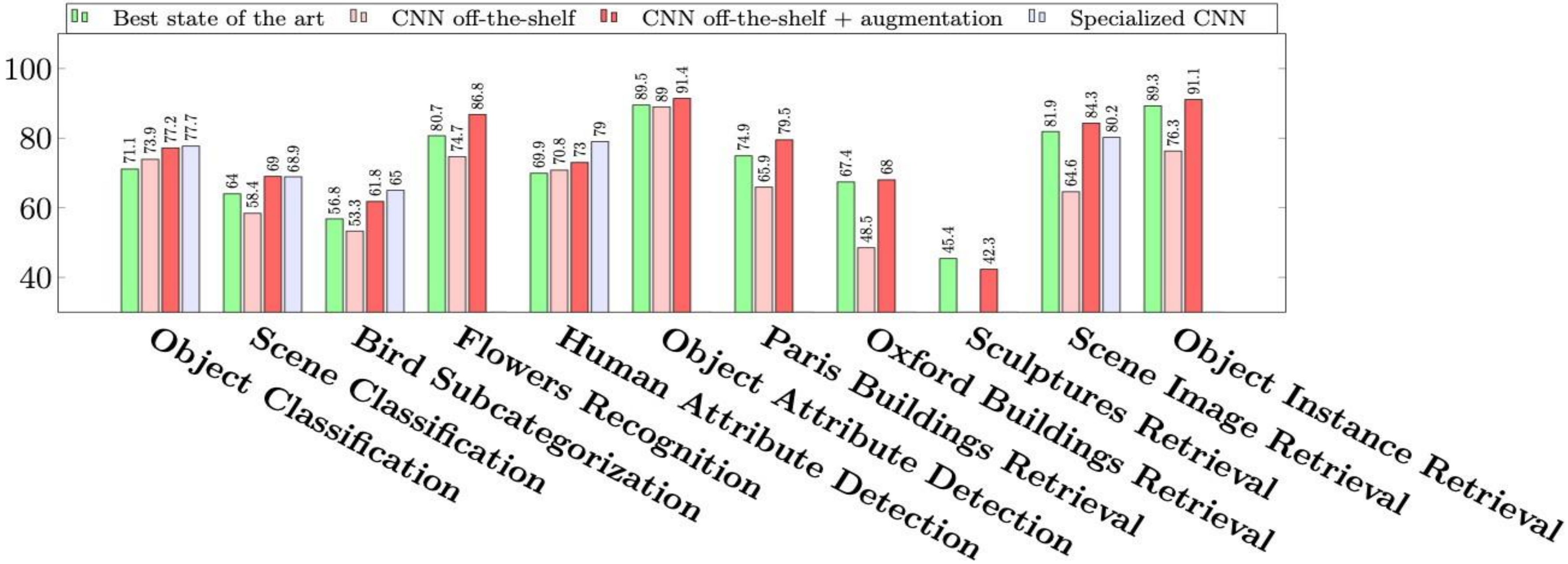
Visualizing Convolutional Networks

Image representation (embedding) from the last layers

- Important observation: the output of final layers serve as good feature representations of images
- For instance, they can be used in tasks like searching for similar images

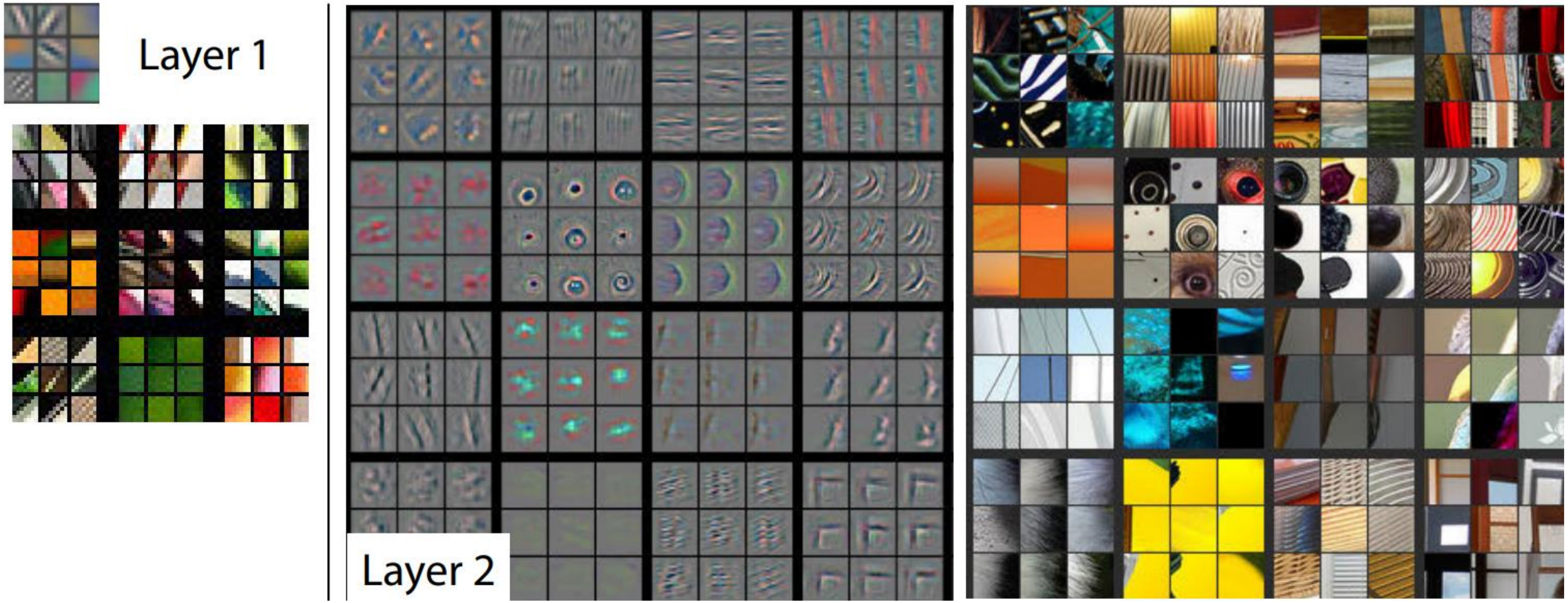
Last layer embeddings





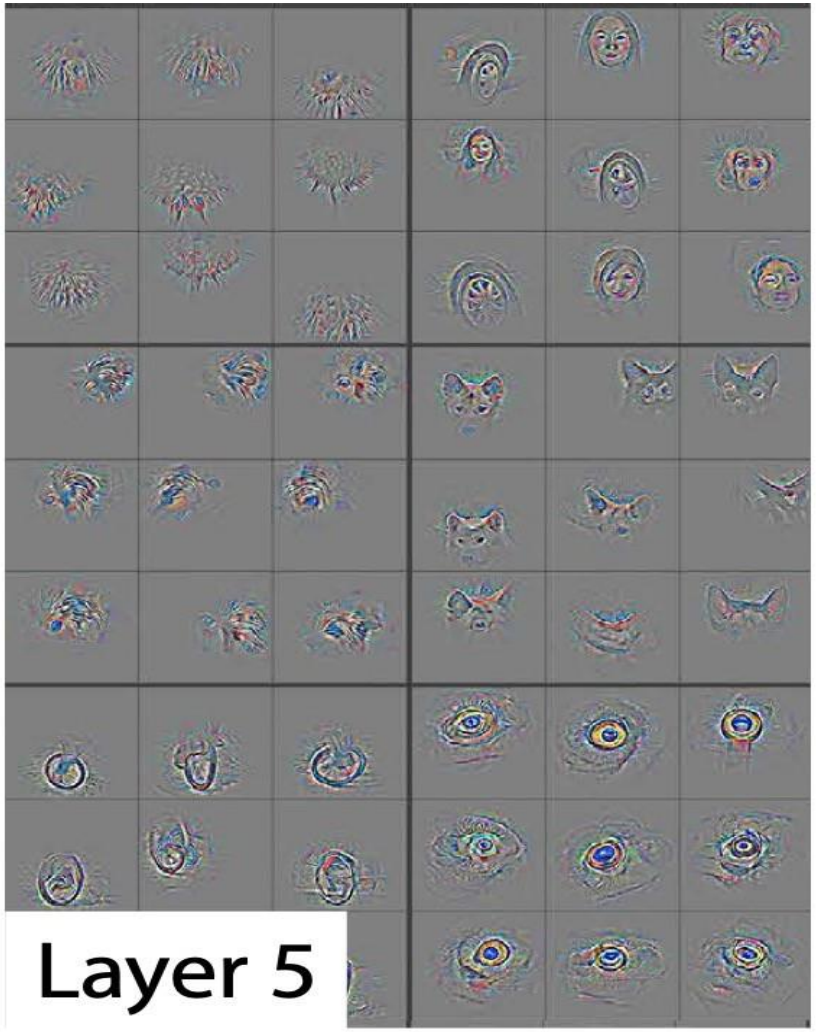
https://arxiv.org/abs/1403.6382

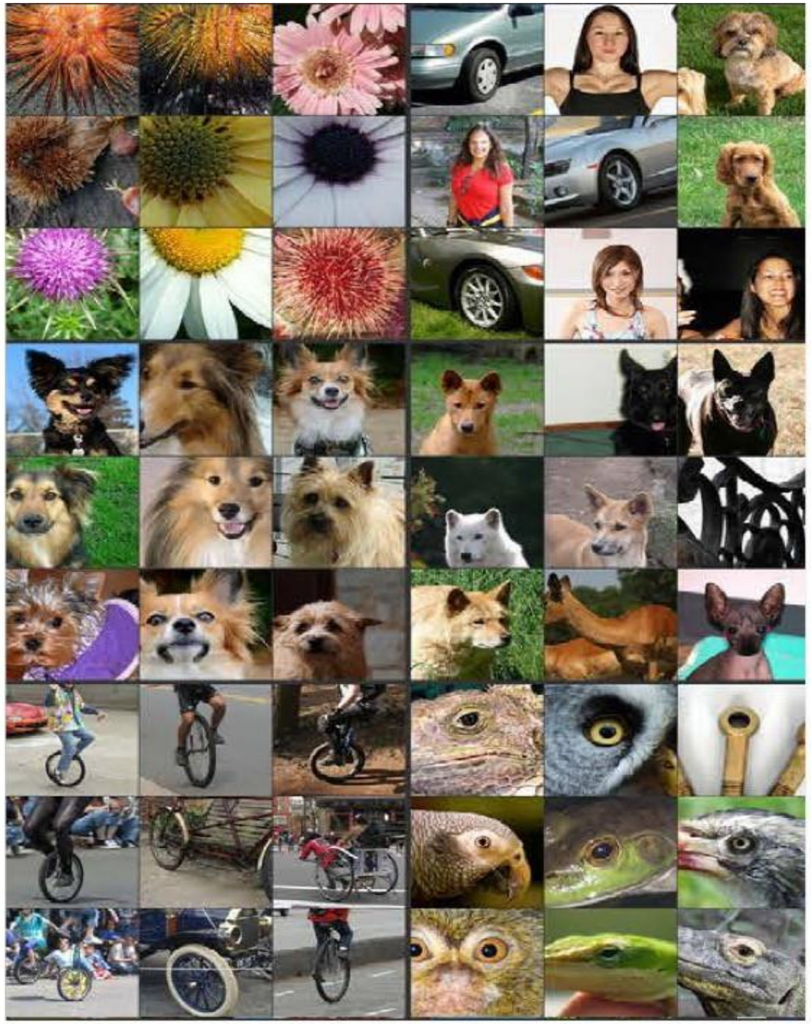
Visualizing Convolutional Networks



https://arxiv.org/abs/1311.2901

Visualizing Convolutional Networks





https://arxiv.org/abs/1311.2901